In [ ]:
```
import pandas as pd
```

In [ ]:
```
df = pd.read_csv("breastcancer.csv")
df.head(10)
```

Out[ ]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concav
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	
5	843786	M	12.45	15.70	82.57	477.1	0.12780	0.17000	
6	844359	M	18.25	19.98	119.60	1040.0	0.09463	0.10900	
7	84458202	M	13.71	20.83	90.20	577.9	0.11890	0.16450	
8	844981	M	13.00	21.82	87.50	519.8	0.12730	0.19320	
9	84501001	M	12.46	24.04	83.97	475.9	0.11860	0.23960	

10 rows × 33 columns

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
Unnamed: 32              569
dtype: int64
```

In [ ]:
```
x = df.iloc[:, 2:-1]
y = df['diagnosis']
```

encoding benign cancer as 0 and malignant cancer as 1

In [ ]:
```
y  = y.map({"B":0,"M":1})
```

Applying Standard Scaling

In [ ]:
```
from sklearn.preprocessing import StandardScaler
```

In [ ]:
```
sc = StandardScaler()
x = sc.fit_transform(x)
```

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state = 134)
```

Importing

In [ ]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

Training a Keras Sequential model using dense layers

Model 1

In [ ]:
```
model1 = Sequential()
model1.add(Dense(32, input_shape = (30,), activation = "relu"))
model1.add(Dense(32, activation = "relu"))
model1.add(Dense(1, activation = "sigmoid"))
model1.summary()
```

```
Model: "sequential_27"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_124 (Dense)           (None, 32)                992

 dense_125 (Dense)           (None, 32)                1056

 dense_126 (Dense)           (None, 1)                 33

=================================================================
Total params: 2,081
Trainable params: 2,081
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
model1.compile(optimizer = "sgd", loss = "binary_crossentropy", metrics = ["accuracy"])
model1.fit(x_train, y_train, epochs = 10)
```

```
Epoch 1/10
15/15 [==============================] - 0s 1ms/step - loss: 0.6213 - accuracy: 0.6571
Epoch 2/10
15/15 [==============================] - 0s 2ms/step - loss: 0.5346 - accuracy: 0.7495
Epoch 3/10
15/15 [==============================] - 0s 2ms/step - loss: 0.4753 - accuracy: 0.8154
Epoch 4/10
15/15 [==============================] - 0s 2ms/step - loss: 0.4272 - accuracy: 0.8791
Epoch 5/10
15/15 [==============================] - 0s 2ms/step - loss: 0.3889 - accuracy: 0.9055
Epoch 6/10
15/15 [==============================] - 0s 2ms/step - loss: 0.3561 - accuracy: 0.9187
Epoch 7/10
15/15 [==============================] - 0s 2ms/step - loss: 0.3271 - accuracy: 0.9319
Epoch 8/10
15/15 [==============================] - 0s 2ms/step - loss: 0.3024 - accuracy: 0.9341
Epoch 9/10
15/15 [==============================] - 0s 2ms/step - loss: 0.2811 - accuracy: 0.9363
Epoch 10/10
15/15 [==============================] - 0s 2ms/step - loss: 0.2620 - accuracy: 0.9407
```

Out[ ]: <keras.callbacks.History at 0x7efc50159690>

Model 2

In [ ]:
```
model2 = Sequential()
model2.add(Dense(64, input_shape = (30,), activation = "sigmoid"))
model2.add(Dense(64, activation = "sigmoid"))
model2.add(Dense(32, activation = "sigmoid"))
model2.add(Dense(32, activation = "sigmoid"))
model2.add(Dense(1, activation = "sigmoid"))
model2.summary()
```

```
Model: "sequential_28"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_127 (Dense)           (None, 64)                1984

 dense_128 (Dense)           (None, 64)                4160

 dense_129 (Dense)           (None, 32)                2080

 dense_130 (Dense)           (None, 32)                1056

 dense_131 (Dense)           (None, 1)                 33

=================================================================
Total params: 9,313
Trainable params: 9,313
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
model2.compile(optimizer = "adam", loss = "binary_crossentropy", metrics = ["accuracy"])
model2.fit(x_train, y_train, epochs = 20)
```

```
Epoch 1/20
15/15 [==============================] - 1s 2ms/step - loss: 0.6885 - accuracy: 0.6308
Epoch 2/20
15/15 [==============================] - 0s 2ms/step - loss: 0.6514 - accuracy: 0.6308
Epoch 3/20
15/15 [==============================] - 0s 2ms/step - loss: 0.6447 - accuracy: 0.6308
Epoch 4/20
15/15 [==============================] - 0s 2ms/step - loss: 0.6291 - accuracy: 0.6308
Epoch 5/20
15/15 [==============================] - 0s 2ms/step - loss: 0.5982 - accuracy: 0.6308
Epoch 6/20
15/15 [==============================] - 0s 2ms/step - loss: 0.5402 - accuracy: 0.6308
Epoch 7/20
15/15 [==============================] - 0s 2ms/step - loss: 0.4537 - accuracy: 0.7451
Epoch 8/20
15/15 [==============================] - 0s 2ms/step - loss: 0.3538 - accuracy: 0.9473
Epoch 9/20
15/15 [==============================] - 0s 2ms/step - loss: 0.2714 - accuracy: 0.9604
Epoch 10/20
15/15 [==============================] - 0s 2ms/step - loss: 0.2107 - accuracy: 0.9692
Epoch 11/20
15/15 [==============================] - 0s 2ms/step - loss: 0.1689 - accuracy: 0.9780
Epoch 12/20
15/15 [==============================] - 0s 2ms/step - loss: 0.1454 - accuracy: 0.9824
Epoch 13/20
15/15 [==============================] - 0s 2ms/step - loss: 0.1232 - accuracy: 0.9846
Epoch 14/20
15/15 [==============================] - 0s 2ms/step - loss: 0.1106 - accuracy: 0.9846
Epoch 15/20
15/15 [==============================] - 0s 2ms/step - loss: 0.1009 - accuracy: 0.9846
Epoch 16/20
15/15 [==============================] - 0s 1ms/step - loss: 0.0934 - accuracy: 0.9846
Epoch 17/20
15/15 [==============================] - 0s 2ms/step - loss: 0.0900 - accuracy: 0.9802
Epoch 18/20
15/15 [==============================] - 0s 2ms/step - loss: 0.0843 - accuracy: 0.9846
Epoch 19/20
15/15 [==============================] - 0s 2ms/step - loss: 0.0819 - accuracy: 0.9868
Epoch 20/20
15/15 [==============================] - 0s 2ms/step - loss: 0.0757 - accuracy: 0.9868
```

Out[ ]: <keras.callbacks.History at 0x7efc8df0a0b0>

Model 1 gives an accuracy of 92.11 %

In [ ]:
```
model1.evaluate(x_test, y_test)[1]
```

```
4/4 [==============================] - 0s 1ms/step - loss: 0.2740 - accuracy: 0.9211
```

Out[ ]: 0.9210526347160339

Model 2 gives an accuracy of 98.25 %

In [ ]:
```
model2.evaluate(x_test, y_test)[1]
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.0904 - accuracy: 0.9825
```

Out[ ]: 0.9824561476707458